AMENDMENTS TO THE CLAIMS

1. (Original) A method for use by a host microprocessor which translates sequences of instructions from a target instruction set for a target processor to sequences of instructions for the host microprocessor comprising the steps of:

beginning execution of a first sequence of target instructions by committing state of the target processor and storing memory stores generated by previously-executed sequences of instructions at a point in the execution of instructions at which state of the target processor is known,

beginning execution of a speculative sequence of host instructions following a branch from the first sequence of target instructions by immediately committing state and storing memory stores,

attempting to execute the speculative sequence of host instructions until another point in the execution of target instructions at which state of the target processor is known,

rolling back to last committed state of the target processor and discarding memory stores generated by the speculative sequence of host instructions if execution fails, and beginning execution of a next sequence of target instructions if execution succeeds.

Claim 2. (Original) A method as claimed in Claim 1 including an additional step of releasing a lock for any sequence of host instructions running in a locked condition immediately after committing state of the target processor and storing memory stores generated by previously-executed translation sequences.

Claim 3. (Cancelled)

Claim 4. (Previously Presented) A method for use by a host microprocessor which translates sequences of instructions from a target instruction set for a target processor to sequences of instructions for the host microprocessor comprising the steps of:

translating a first speculative sequence of host instructions from a first sequence of target instructions from a point in the translation of target instructions at which state of the target processor is known,

ending the first sequence of target instructions in response to encountering a branch from the first sequence in the target program by:

branching to a branch sequence of target instructions,
committing state of the target processor and storing memory stores generated by the first translation sequence after a branch taken before executing a branch sequence of host instructions, and

ending execution of the first sequence of target instructions if a branch is not taken from the first sequence by:

rolling back to last committed state of the target processor and discarding memory stores generated by the speculative sequence of host instructions if execution fails, and committing state of the target processor and storing memory stores generated by the first sequence at the end of the sequence of target instructions at which state of the target processor is known.

5. (NEW) A method for use by a host microprocessor which translates sequences of instructions from a target instruction set a target processor to sequences of instructions for the host microprocessor comprising the steps of:

beginning execution of a first sequence of target instructions by committing state of the target processor and storing memory stores generated by previously executed sequences of instructions at a point in execution of instructions at which state of the target processor is known;

executing a sequence of host instructions from the first sequence of target instructions commencing immediately after committing state of the target processor and storing memory stores previously generated by execution until another point in the translation of target instructions at which state of the target processor is known;

beginning execution of a next sequence of target instructions by committing state of the target processor and storing memory stores generated by the execution of the first sequence of target instructions at the another point in the translation of target instructions at which state of the target processor is known; and

executing a sequence of host instructions from the next sequence of target instructions commencing immediately after committing state of the target processor and storing memory stores generated by execution of the first sequence of target instructions until a further point in the translation of target instructions at which state of the target processor is known.

6. (New) A method as claimed in Claim 5 including an additional step of releasing a lock for any sequence of host instructions running in a locked condition immediately after committing state of the target

processor and storing memory stores generated by previously-executed translation sequences.

7. (New) A method of executing instructions in a host microprocessor which translates sequences of instructions from a target instruction set for a target processor to sequences of instructions for the host microprocessor, said method comprising:

speculating that execution of a branch instruction will cause a branch to a first sequence of host instructions;

committing state of the target processor and storing memory stores from previously executed host instructions after the branch instruction is executed and prior to beginning execution of the first sequence of host instructions; and

executing the first sequence of host instructions.

8. (NEW) A method of executing instructions in a host microprocessor which translates sequences of instructions from a target instruction set for a target processor to sequences of instructions for the host microprocessor, said method comprising:

if a sequence of instructions includes a locking operation, placing a commit operation at the beginning of the sequence of instructions including the locking operation; and if a sequence of instructions does not include a locking operation, placing a commit operation at the end of the sequence of instructions not including the locking operation.